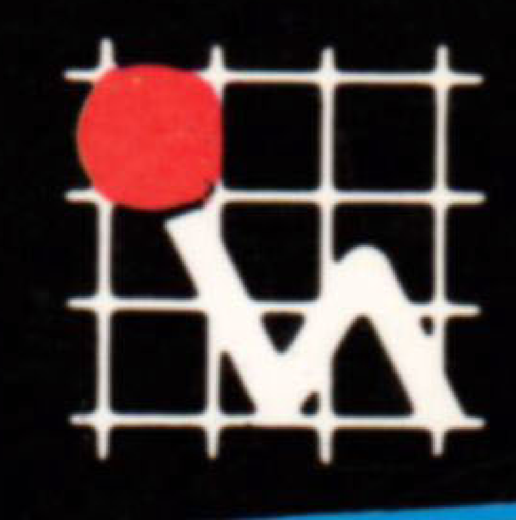
SONY

SONY

MSX



ENSAMBLADOR DESENSAMBLADOR

DEVPAC

DEVPAC

Edición original: HISOFT. Primera edición: 1985.
Edición española: INDESCOMP, S.A.

HISOFT, 180 High Street, North Dunstable LU6 1AT
Indescomp, S.A., Avda. del Mediterráneo 9, 28007 Madrid

Producción de la edición española:
Vector Ediciones, Gutierre de Cetina 61, 28017 Madrid

Traducción:
Jesús Rojo García
Profesor de Matemática Aplicada
Escuela T. Sup. de Ing. Industriales de Valladolid

Impresión:
Gráficas Lormo, Isabel Méndez 15, 28038 Madrid

CONTENIDO

PRIMERA PARTE ENSAMBLADOR/EDITOR GEN

SECCIÓN 1 CÓMO EMPEZAR **6**

SECCIÓN 2 EL ENSAMBLADOR **7**

2.0 Cómo trabaja GEN 7
2.1 Formato de las sentencias del ensamblador 9
2.2 Etiquetas 10
2.3 Contador de posición 10
2.4 Tabla de símbolos 10
2.5 Expresiones 11
2.6 Seudo-operaciones 12
2.7 Seudo-operaciones de tipo condicional 13
2.8 Comandos del ensamblador 14
2.9 Macros 15
2.10 GEN y el mapa de memoria de MSX 17

SECCIÓN 3 EL EDITOR **18**

3.1 Introducción al editor 18
3.2 Comandos del editor 19
3.2.1 Creación e inserción de texto 19
3.2.2 Listado de texto 20
3.2.3 Edición de texto 20
3.2.4 Grabación y carga de ficheros 21
3.2.5 Ensamblado y ejecución 23
3.2.6 Otros comandos 23
3.3 Un ejemplo sencillo de utilización del editor 24

APÉNDICE 1 MENSAJES DE ERROR **26**

APÉNDICE 2 PALABRAS RESERVADAS, CÓDIGOS, ETC. **27**

APÉNDICE 3 UN EJEMPLO DESARROLLADO **28**

SEGUNDA PARTE DESENSAMBLADOR/DEPURADOR MON

SECCIÓN 1 INTRODUCCIÓN **34**

1.1 Carga del programa 34
1.2 Lo que aparece en la pantalla 34

SECCIÓN 2 COMANDOS HABITUALES **37**

2.1 Introducción de números 37
2.2 Descripción de los comandos 37
2.3 Cuadro resumen 41

SECCIÓN 3 COMANDOS AVANZADOS **43**

3.1 Más sobre cómo cargar el programa 43
3.2 Comandos para desensamblar 43
3.3 Comandos de interrupción 45
3.4 Comados de ejecución 46
3.5 Otro comando 47

PRIMERA PARTE

GEN

SECCIÓN 1. CÓMO EMPEZAR

GEN es un ensamblador para el Z8Ø, potente, de utilización sencilla y muy semejante al ensamblador estándar de Zilog. A diferencia de otros ensambladores para microordenador, GEN es un programa con una gama muy amplia de posibilidades, y será conveniente que, antes de comenzar a utilizar el ensamblador, estudie las secciones que siguen así como el ejemplo del apéndice 3. Si es usted principiante en el tema, le aconsejamos que comience por trabajar con el ejemplo del apéndice 3.

GEN posee unos 78ØØ bytes de longitud; además, su editor coloca a continuación de GEN el fichero de texto que se crea y después la tabla de símbolos del ensamblador. En consecuencia, cuando cargue GEN debe pensar en dejar suficiente espacio además para el fichero de texto y la tabla de símbolos que piensa crear en la sesión de trabajo. Puede ser conveniente cargar GEN en lugares bajos de la memoria.

GEN se puede colocar en cualquier parte de la memoria de su ordenador MSX en la que no entre en conflicto con el espacio que utiliza el sistema operativo; esto significa que se lo puede colocar entre las direcciones 128 y la #C8ØØ (para una máquina con 64K). Si utiliza disco, no debe cargar GEN en una posición tan alta como #C8ØØ, ya que el software del lector de discos ocupa posiciones de dicho área de memoria.

Para cargar GEN desde BASIC, utilice el comando:

RUN "CAS:GENMSX" ⟨RETURN⟩

Con esto cargará el ensamblador en la dirección #81ØØ y lanzará automáticamente la ejecución.

Si desea cargar GEN en una dirección diferente (para máquinas de 64K la dirección 128, o sea #8Ø, puede ser buena), debe hacer lo siguiente:

BLOAD "CAS:GENMSX" ⟨RETURN⟩

que carga el programa en #88ØØ, y después

DEFUSR = &H88ØØ ⟨RETURN⟩
A = USR(X) ⟨RETURN⟩

donde X es la dirección en la que desea que comience GEN. El programa se colocará automáticamente en esa posición y se pondrá en modo de edición.

La sección 2 describe el ensamblador de GEN; la sección 3, el editor; y el apéndice 3 desarrolla un ejemplo completo.

SECCIÓN 2. EL ENSAMBLADOR

2.0 Cómo trabaja GEN

GEN es un ensamblador para Z80, fácilmente adaptable a la mayor parte de los sistemas que poseen dicho microprocesador. Como aspectos distintivos puede decirse de él que realiza dos pasadas (la primera de las cuales la utiliza para buscar errores y catalogar los símbolos), que puede ensamblar todos los códigos de operación nemotécnicos del Z80, que posee muchos comandos específicos, que permite definir macro-instrucciones, que admite el ensamblado condicional y que crea una tabla de símbolos de consulta muy rápida.

Cuando se utiliza el comando 'A' del editor (véase la sección 3), que sirve para ensamblar un fichero de texto ('A' de "assembly"), el programa comienza por pedir, mediante el mensaje 'Table size:', la cantidad de espacio que hay que reservar para la tabla de símbolos que se genera durante el ensamblado. Su respuesta debe ser un número decimal terminado por ⟨RETURN⟩. Puede también dejar que el programa decida esta cantidad pulsando simplemente ⟨RETURN⟩; entonces, GEN escoge un tamaño para la tabla adecuado al del tamaño del texto. Observe que, si va a utilizar el comando '*F' para incluir texto durante el ensamblado, deberá prever un tamaño más grande de lo normal, y en ese caso el ensamblador no puede predecir el tamaño del fichero que va a ser incluido.

Después del tamaño de la tabla, se le pedirá que especifique las opciones de ensamblado que desea mediante el mensaje 'Options:'. La respuesta debe ser el número decimal que resulta de sumar los números correspondientes a las opciones que se deseen. Las opciones posibles son:

- opción 1 realiza un listado de la tabla de símbolos al final de la segunda pasada de ensamblado;
- opción 2 no genera código objeto;
- opción 4 no realiza el listado de ensamblador;
- opción 8 envía el listado de ensamblador a la impresora;
- opción 16 coloca el código objeto, si se genera, después de la tabla de símbolos; es decir, va actualizando el contador de posición al encontrar una seudo-operación ORG, pero no coloca el código en el lugar donde debe ejecutarse;
- opción 32 suprime toda comprobación acerca de dónde se sitúa el código objeto; suele ser útil para que el ensamblado se realice de manera más rápida.

Por ejemplo, la utilización de 36 como opción (36 = 4 + 32), proporciona un ensamblado rápido, sin listado y sin comprobaciones sobre la posición en que se coloca el código objeto.

Obsérvese que, si se utiliza la opción 16, la seudo-operación ENT quedará sin efecto. Por otra parte, aunque haya utilizado esta opción, puede conseguir saber donde ha sido colocado el código objeto; para ello basta que averigüe la posición del final del fichero de texto (con el comando 'X' del editor) y sume a dicha posición la longitud de la tabla de símbolos más 2.

Para llevar a cabo el ensamblado se realizan dos pasadas. En la primera, GEN localiza los errores y compila la tabla de símbolos. En la segunda se genera el código objeto (salvo que se esté utilizando la opción 2). Durante la primera pasada no hay salidas a la pantalla ni a la impresora, salvo que se detecte algún error. En ese caso aparece en la pantalla la línea errónea y bajo ella el correspondiente

número de error (de entre los que figuran en el apéndice 1); además, el proceso de ensamblado se detiene. Pulse ⟨CTRL/STOP⟩ si desea entonces volver al editor, o cualquier otra tecla si quiere que el ensamblado continúe a partir de la siguiente línea.

Al terminar la primera pasada aparece el aviso

Pass 1 errors: nn (se ha concluido la primera pasada y se han encontrado nn errores).

Si se ha detectado algún error, el ensamblado se detiene y no se produce la segunda pasada. Si se han utilizado como operandos etiquetas ("labels") que no hayan sido definidas, aparecerá el mensaje '*WARNING*' label absent' (*aviso*...) tantas veces como etiquetas falten por definir.

El código objeto se genera durante la segunda pasada salvo, como se ha dicho, si se utiliza la opción 2. También se genera el listado de ensamblador, a menos que se haya utilizado la opción 4 o el comando '*L –' (en la sección 2.8 se pueden ver los comandos del ensamblador). Cada línea normal de este listado es de la forma

```
C000  210100       25 etiqueta:   LD   HL,1
1     7        16   22            32   37
```

Como se puede observar, cada línea está dividida en campos que comienzan en las posiciones 1, 7, ... que se indican. El primer campo es el valor del contador de posición en el momento en que se comienza a procesar la línea, a menos que el código nemotécnico de la línea sea una de las seudo-operaciones ORG, EQU o ENT (véase la sección 2.6). El valor se representa normalmente en hexadecimal, a menos que se utilice el comando '*D+' (véase la sección 2.8), en cuyo caso se lo representa en notación decimal.

El siguiente campo, que comienza en la columna 7, es el código objeto correspondiente a la línea. Su máxima longitud es de 8 cifras hexadecimales, con lo que puede representar hasta 4 bytes.

A continuación viene el número de línea, un número entero entre 1 y 65535 inclusive.

Si en la línea se ha definido una etiqueta, los primeros 8 caracteres de esta etiqueta aparecen en el campo que comprende las columnas 22 a 31. Si la etiqueta es más larga y desea que aparezca enteramente, defínala en una línea aparte, como por ejemplo

```
etiqueta_muy_larga:
      LD HL,0
```

El código de operación nemotécnico aparece en las columnas 32 a 35. Los posibles operandos figuran a partir de la columna 37 y los comentarios, si los hay, a partir de la columna 55.

Se puede utilizar el comando '*C' para generar un listado más compacto, lo que puede ser conveniente cuando el listado se envía a la pantalla; la opción '*C –' de dicho comando suprime las 9 columnas del campo correspondiente al código objeto y corre 9 posiciones hacia la izquierda el resto del listado.

La anchura de pantalla que utiliza el programa GEN está almacenada en la posición 'comienzo de GEN+30'; habitualmente la anchura es de 37 columnas.

Puede detener el listado pulsando ⟨CTRL/STOP⟩; a continuación puede pulsar de nuevo ⟨CTRL/STOP⟩ si desea volver al editor, o cualquier otra tecla si desea que el listado continúe.

Los únicos errores que pueden aparecer durante la segunda pasada son los que se señalan con '*ERROR* 10' (véase el apéndice 1) y con 'Bad ORG!' (que ocurre cuando se han asignado al código objeto posiciones que lo colocan sobre el propio programa GEN, el fichero de texto o la tabla de símbolos; esto puede no detectarse si se ha utilizado la opción 32). Si aparece '*ERROR* 10' se puede continuar con el ensamblado en la misma manera que explicamos para la primera pasada, pero si aparece 'Bad ORG!' el programa pasa automáticamente al editor.

Al terminar la segunda pasada aparecerá el mensaje:

Pass 2 errors: nn

acompañado tal vez de avisos sobre las etiquetas que no se han definido, de la misma manera que en la primera pasada. A continuación aparece el mensaje

Table used: xxxxx from yyyyy   (parte usada de la tabla: xxxxx de yyyyy)

que le informa sobre la porción utilizada del espacio que se reservó para la tabla de símbolos.

A continuación, si se ha usado correctamente la seudo-operación ENT, aparecerá en pantalla

'Executes: nnnnn'

que indica la dirección que se debe llamar para comenzar la ejecución del código objeto. La ejecución se puede ordenar con el comando 'R' del editor. No se debe utilizar este comando si no se ha realizado satisfactoriamente el ensamblado y aparecido el mensaje 'Execute: nnnnn'.

Finalmente, si se ha utilizado la opción 1, aparecerá al final del listado una lista de las etiquetas usadas y de sus valores asociados. La lista tendrá dos etiquetas en cada línea salvo que se cambie esta cantidad, que figura como parámetro en la posición 'comienzo de GEN + 27'.

Terminado este proceso, el control vuelve al editor.

2.1 Formato de las sentencias del ensamblador

Las líneas que debe procesar el ensamblador GEN han de responder al siguiente formato, algunos de cuyos campos son opcionales:

ETIQUETA	CODIGO	OPERANDOS	COMENTARIO
arranque:	LD	HL,simbolo	;se carga 'simbolo'

Se ignoran los espacios y los tabuladores introducidos por el editor.

Cada línea es procesada de la forma que explicamos a continuación.

Si el primer carácter que se encuentra en una línea (salvo los blancos) es alguno de los símbolos ';', '*' o 'CR' (este último es el carácter llamado retorno de carro o fin de línea, y se obtiene mediante la tecla ⟨RETURN⟩), el ensamblador toma las siguientes decisiones según el símbolo de que se trate:

';' la línea entera se considera un comentario, lo que equivale de hecho a ignorarla;

'*' considera el carácter siguiente, o los siguientes, como uno de los comandos que se explican en la sección 2.8; los caracteres posteriores al comando se consideran como un comentario, salvo en el caso de los comandos '*F' y '*H';

'CR' se ignora la línea.

Si el primer carácter es diferente de éstos, el ensamblador comienza por recorrer la línea en busca del símbolo ':'. Si lo encuentra, todos los caracteres anteriores a él son interpretados como definición de una etiqueta.

A continuación, si el siguiente carácter que encuentra no indica fin de línea o comienzo de un comentario, trata de interpretarlo como el comienzo de un código de operación nemotécnico, código que puede tener hasta 4 caracteres de longitud y que debe terminar con un espacio en blanco si la línea continúa. Si el código encontrado es válido y requiere operandos, el ensamblador los busca a continuación. Los comentarios se deben situar después de los operandos, salvo que el código de operación no los necesite, en cuyo caso se pueden poner los comentarios a continuación del código de operación.

Puede haber líneas que consistan exclusivamente en una etiqueta; a veces esto es útil para que el listado sea más claro. Tanto el editor como el ensamblador truncan por sus 8 primeros caracteres las etiquetas demasiado largas, a no ser que la línea esté dedicada exclusivamente a definir la etiqueta.

2.2 Etiquetas

Una etiqueta (''label'') es un símbolo que se emplea para representar ciertos datos cuya longitud sea como máximo de 16 bits. Puede representar la dirección de una instrucción o de un área de datos. También se la puede utilizar como constante mediante la seudo-operación EQU (véase la sección 2.6).

La definición de una etiqueta (o sea, el nombre) debe ir seguida del símbolo ':'. En el nombre se pueden utilizar los caracteres Ø . . . 9, $ y A . . . z, lo que incluye, además de las letras mayúsculas y minúsculas, los símbolos [, \,], ↑, ' y _; el primer carácter debe ser obligatoriamente una letra. El nombre puede tener cualquier longitud, pero sólo se consideran significativos sus 8 primeros caracteres. Si los 8 primeros caracteres de dos etiquetas coinciden, se produce la redefinición de una misma etiqueta, lo que es ilegal y produce el mensaje de '*ERROR* 4'. El nombre de una etiqueta no puede ser una de las palabras reservadas, cuyo catálago se encuentra en el apéndice 2; sin embargo, una de estas palabras puede formar parte del nombre de una etiqueta. Ya hemos dicho que el editor trunca por sus 8 primeros caracteres las etiquetas demasiado largas, salvo que la línea se dedique exclusivamente a definir la etiqueta. Todos los nombres que ponemos a continuación son válidos:

```
BUCLE:
bucle:
etiqueta_muy_larga:
L[1]:
a:
LDIR:     (LDIR es un código de operación, pero no una palabra reservada)
dos↑5:
```

Cuando una etiqueta que representa un valor que ocupa más de 8 bits se utiliza en un contexto en el que se requiere una constante de 8 bits, el ensamblador detecta un error (el '*ERROR* 1Ø') durante la segunda pasada. Es lo que ocurriría con el ejemplo siguiente:

```
etiqueta:  EQU  #1234
           LD   A,etiqueta
```

2.3 Contador de posición

El ensamblador mantiene un contador de posición (''location counter'') que permite asociar a las etiquetas direcciones de la memoria y reflejar esta correspondencia en la tabla de símbolos. Para dar un valor determinado al contador de posición se emplea la seudo-operación ORG (véase la sección 2.6).

El símbolo '$' se puede utilizar para representar el valor que en ese momento tenga el contador de posición. Así, por ejemplo, el código generado por la sentencia LD HL,$+5 cargará el par de registros HL con el valor del contador más 5.

2.4 Tabla de símbolos

Cuando el ensamblador encuentra una nueva etiqueta, la coloca en un archivo de símbolos que crea, denominado *tabla de símbolos* (''symbol table''). Con cada etiqueta coloca el valor de la etiqueta y dos punteros. Si el ensamblador ha encontrado la etiqueta en la línea en la que se define, el valor que le asigna es el que tiene el contador de posición en ese momento o bien, si se la define mediante la seudo-operación EQU, el que tenga la correspondiente expresión que sigue a la operación. Puede ocurrir que el ensamblador encuentre por primera vez una etiqueta antes de su definición; entonces espera a encontrarla definida antes de asignarle un valor.

Por su parte, los dos punteros se ocupan de relacionar la etiqueta con la anterior y la posterior en orden alfabético. Esto da a la tabla de símbolos una estructura del tipo llamado ''de árbol binario'', que es

un sistema rápido para la localización de datos, aspecto esencial cuando se ensambla un programa largo.

El espacio que ocupa un símbolo en la tabla varía entre 8 y 15 bytes, según sea de largo el nombre. Sólo los 8 primeros caracteres del nombre se guardan en la tabla, aunque éste sea más largo.

Cuando un símbolo se define más de una vez, se produce una indeterminación en cuanto al valor que debe tomar; esto produce un mensaje de error ('*ERROR* 4') durante la primera pasada del ensamblador.

Cuando el valor de un símbolo no se define, también se genera un mensaje de error al final del ensamblado ('*WARNING* symbol absent'), aunque en este caso el error no impide la continuación del ensamblado.

Al final del proceso de ensamblado se emite un mensaje que indica la cantidad de memoria que ha ocupado la tabla de símbolos. Como se dijo en la sección 2.Ø, la opción 'Table:' permite cambiar la cantidad de memoria que se asigna a la tabla de símbolos.

2.5 Expresiones

Las expresiones que se pueden poner como operandos de las distintas operaciones y seudo-operaciones pueden consistir en un simple *término* o en varios *términos* separados por *operadores*. Por *término* y *operador* se entiende lo que explicamos a continuación.

Un *término* es alguna de las siguientes cosas:

un número decimal, como 1Ø29, por ejemplo;
un número hexadecimal, como #4Ø5;
un número binario, como %1ØØØØØØØ1Ø1;
una constante literal, como "a";
una etiqueta, como Etiqueta
el símbolo $, en representación del valor del contador de posición.

Un *operador* es alguno de los siguientes símbolos, cuyo significado se explica a continuación:

'+' suma;
'–' resta;
'&' el AND logico;
'@' el OR lógico;
'!' el XOR lógico;
'*' multiplicación entera;
'/' división entera;
'?' la función MOD (a?b = a – (a/b)*b)

Presentamos a continuación algunos ejemplos de expresiones válidas, junto con el valor que poseen cuando éste se puede calcular:

#5ØØØ – etiqueta	
%1ØØ11Ø1 ! %1Ø11	que vale %1ØØØ11Ø
#3456 ? #1ØØØ	que vale #456
4 + 5 * 3 – 8	que vale 19
$ – etiqueta + 8	
2345/7 – 1	que vale 334
"A" + 128	
"y" – ";" + 7	
(5 * etiqueta – #1ØØØ & %1111)	
17 @ %1ØØØ	que vale 25

Nótese que se puede insertar blancos entre términos y operadores, pero no en medio de un término.

Se utilizan los símbolos '#' y '%' para señalar los números hexadecimales y binarios, y las dobles comillas para señalar las constantes literales. Cuando el ensamblador lee un número de cualquier tipo, lo reduce a sus 16 bits menos significativos, esto es, lo toma módulo 65536. Por ejemplo, el número 7ØØ16 se convierte en 448Ø y el número #5A2C4 en #A2C4.

El orden en que se evalúan las expresiones es estrictamente de izquierda a derecha, sin que exista ninguna jerarquía entre los operadores. Ciertas conveniencias han aconsejado dotar al programa de los operadores '*', '/' y '?', lo que no significa que se hayan previsto todas las posibilidades de manejo de expresiones; tal propósito hubiese hecho demasiado grande el tamaño de GEN.

Las expresiones que se encierran entre paréntesis representan direcciones de memoria. La instrucción LD HL,(loc+5) servirá para cargar en el par de registros HL los 16 bits contenidos en las posiciones de memoria loc+5 y siguiente.

Ciertas instrucciones del Z8Ø, como JR y DJNZ, funcionan mediante direccionamiento relativo, que viene dado por un operando de 8 bits. Cuando el ensamblador se encuentra con una de estas sentencias, obtiene el salto relativo restando del valor que figura en el operando de la instrucción el valor del contador de posición de la siguiente instrucción. Luego los valores que se pueden poner en el operando están en el intervalo de −128 a 127 alrededor del valor del contador de posición de la instrucción siguiente. O, lo que es lo mismo, están en el intervalo de −126 a 129 alrededor del valor del símbolo $ en la propia instrucción.

Si una multiplicación produce un resultado cuyo valor absoluto sobrepasa 32767, se obtiene el mensaje '*ERROR* 15'. La división por cero produce el mensaje '*ERROR* 14'; si se produce sobrepasamiento ("overflow") en la división, no se genera ningún aviso. Se utiliza la representación de complemento a 2, lo que hace que los números mayores que 32767 sean tratados como negativos; por ejemplo, 6ØØØØ = −5536 (= 6ØØØØ − 65536).

2.6 Seudo-operaciones

Además de los códigos nemotécnicos correspondientes a las operaciones propias del Z8Ø, el ensamblador GEN reconoce otra serie de códigos nemotécnicos que no corresponden a operaciones y que, por lo tanto, no se plasmarán directamente en códigos binarios de operación al ser ensamblados. Su objetivo es proporcionar al propio ensamblador instrucciones que le permitan realizar ciertas acciones en la forma conveniente, como, por ejemplo, ajustar el contador de posición, dar valores a las etiquetas, etc. Estas acciones reciben el nombre de seudo-operaciones ("pseudo-mnemonics" o también "assembler directives", esto es, instrucciones de ensamblador); poseen códigos nemotécnicos semejantes a los de las operaciones y utilizan también operandos; se escriben de la misma manera que las operaciones, precedidas posiblemente de una etiqueta y seguidas de comentarios.

Algunas seudo-operaciones originan una acción variable dependiendo de ciertos valores; se llaman condicionales y serán tratadas en la sección siguiente. Daremos ahora una lista de las restantes seudo-operaciones, junto con su sintaxis y una descripción de la acción que realizan. La palabra 'expresión' tiene el mismo sentido que en la sección precedente. Aunque no se indique, todas pueden ir precedidas de una etiqueta.

ORG expresión

Hace que el contador de posición tome el valor de la expresión. Si no se están empleando las opciones 2 o 16, ORG es tal que haría que el código objeto se escribiera en parte en las posiciones ocupadas por GEN, por el fichero de texto o por la tabla de símbolos, se interrumpe el ensamblado y aparece el mensaje de error 'Bad ORG!', como ya se explicó en la sección 2.Ø.

Etiqueta EQU expresión

Debe comenzar necesariamente por un nombre de etiqueta. Asigna a dicha etiqueta el valor de la expresión. La expresión no puede llevar símbolos cuyo valor no haya sido definido; en caso contrario aparece el mensaje de error '*ERROR* 13'.

DEFB expresión,expresión,...

Sirve para almacenar valores en posiciones de memoria. El valor de cada expresión debe ocupar un máximo de 8 bits (la 'B' del código significa "byte"). El efecto que se produce es colocar en la posición de memoria cuya dirección es la que indique el contador de posición el valor de la primera expresión. A continuación se incrementa el contador de posición en una unidad y se hace lo mismo con la segunda expresión, y así sucesivamente.

DEFW expresión,expresión,...

Sirve también para almacenar valores en posiciones de memoria, pero ahora la longitud de los valores es de 16 bits (la 'W' del código significa "word" o palabra que es el nombre que designa dicha longitud) y su almacenamiento se produce en dos posiciones consecutivas de memoria y el contador de posición se avanza en dos unidades cada vez. La forma de almacenamiento es la habitual: el byte menos significativo se coloca en la primera posición de memoria y el más significativo en la segunda.

DEFS expresión

Sirve para incrementar el contador de posición en una cantidad igual al valor de la expresión, lo que equivale a reservar esa cantidad de posiciones de memoria. Las correspondientes posiciones de memoria se rellenan con ceros ('S' de "space", espacio en blanco).

DEFM "literal"

Sirve para llenar n posiciones de memoria, a partir de la que indique el contador de posición, con el código ASCII de la expresión literal que figura entrecomillada a continuación ('M' de "message", mensaje). La longitud n de la expresión literal puede llegar hasta 255, pero de hecho está limitada por la longitud de la línea que permite el editor. El primer carácter del operando deben ser las dobles comillas; el final de la expresión literal puede venir marcado por otras dobles comillas o por el caracter 'CR' de fin de línea.

ENT expresión

Selecciona la dirección por la que va a comenzar la ejecución del código objeto, dándole el valor que posea la expresión. Esta dirección es la que emplea el comando 'R' que se describe en la sección 3. Si no se emplea explícitamente la seudo-operación END, el ensamblador no coloca por defecto ninguna dirección de comienzo de ejecución.

2.7 Seudo-operaciones de tipo condicional

Como ya se ha dicho, imponen al ensamblador acciones variables dependiendo del valor de alguna expresión. Permiten al programador incluir o no parte del texto en el proceso de ensamblado. Estas seudo-operaciones son tres y se emplean generalmente de forma combinada.

IF expresión

Hace que se calcule el valor de la expresión; si el resultado es cero, las líneas que siguen no son ensambladas hasta que las seudo-operaciones ELSE o END cambien esta situación. Si el valor de la expresión es diferente de cero, el ensamblado prosigue en la forma habitual.

ELSE

Cambia el valor contrario la situación de ensamblado o no ensamblado. Si la situación era de no ensamblado, las líneas vuelven a ensamblarse a partir de la siguiente. Si la situación era de ensamblado en la forma habitual, no se ensamblan las líneas que siguen al ELSE.

END

Hace que las líneas que siguen sean ensambladas normalmente, fuese cual fuese la situación anterior.

Nota. No se pueden utilizar seudo-operaciones condicionales anidadas unas en otras. Cuando no se respeta esta prohibición, los resultados del ensamblado son impredecibles.

2.8 Comandos del ensamblador

Los comandos del ensamblador tienen por objeto modificar de diversas maneras el listado del ensamblador. No tienen efecto alguno sobre el Z8Ø y, a diferencia de las seudo-operaciones, tampoco tienen efecto alguno sobre el código objeto.

Un comando del ensamblador es una línea del fichero de texto que comienza por el símbolo '*'. A continuación debe haber una letra mayúscula, que es la que especifica el comando de que se trata. Los comandos 'L', 'C' y 'M' exigen además uno de los símbolos '+' o '−' después de la letra, lo que hará que el comando tenga un efecto o su contrario. El resto de la línea se considera como un comentario, sea cual sea su contenido.

Los comandos del ensamblador, salvo el '*F', surten efecto solamente durante la segunda pasada.

Si el ensamblado de una parte del texto no se realiza por indicarlo así alguna seudo-operación condicional, quedan también sin efecto los comandos que se encuentren en dicha parte del texto.

Pasaremos a una descripción de los distintos comandos.

comando *E acción: salto de página

Deja tres líneas en blanco en el listado si éste se realiza por pantalla, o causa un salto de página si el listado se envía a la impresora. Se utiliza para separar el listado en módulos.

comando *H literal acción: pone encabezamientos

Hace que la cadena literal se ponga como encabezamiento del trozo siguiente cada vez que se ejecuta el comando '*E' ('H' proviene de "heading", encabezamiento). El comando '*H' hace además que se ejecute al mismo tiempo el comando '*E'.

comando *S acción: detiene el listado

Detiene el listado en la línea del comando ('S' de "stop"); el listado se reanuda cuando se pulsa cualquier tecla.

comando *L – acción: deja sectores sin listar

Hace que cese el listado a partir de su propia línea.

comando *L + acción: vuelve a listar

Reanuda el listado a partir de su propia línea.

comando *C – acción: listado de tipo compacto

Hace que a partir de la siguiente línea el listado sea de tipo compacto, esto es, sin que aparezca el código objeto generado por la línea; se ahorran así 9 posiciones por línea y se consigue que quepa en una sola línea de la pantalla, lo que hace el listado en pantalla más legible.

comando *C + acción: listado de tipo amplio

Causa la acción contraria, o sea, el listado sin abreviar.

comando *M + acción: lista las macro-instrucciones

Hace que en el listado aparezca el desarrollo de las macro-instrucciones (véase la sección 2.9) cuando se las utiliza.

comando *M – acción: no lista las macro-instrucciones

En el listado aparecen únicamente los nombres de las macro-instrucciones. Es la acción que el ensamblador realiza por defecto.

comando *F nombre acción: incluye un fichero de texto

Es un comando que permite ensamblar un fichero de texto que se encuentre grabado en cinta, sin necesidad de cargarlo totalmente en la memoria. En cada paso se carga un bloque del fichero en un tampón (''buffer'') y se procede a su ensamblado; esto se repite con tantos bloques como sea necesario. Como de hecho sólo se cargan pequeñas cantidades de texto a un tiempo, pueden ensamblarse así ficheros voluminosos.

El nombre del fichero debe ir después de la 'F' y separado de ésta por un espacio. La longitud del nombre puede ser de hasta 6 caracteres.

El comando se puede utilizar también para incluir, en cualquier parte de un texto que se esté ensamblando, un fichero previamente grabado en cinta.

Los ficheros que se desee utilizar con este comando deben haber sido cargados utilizando el comando 'P' del editor.'

Cuando el ensamblador detecta el comando 'F', busca en la cinta un fichero de tipo válido (creado con el comando 'P') y con el nombre adecuado. Entretanto va escribiendo en pantalla el nombre de los ficheros que encuentra antes del que busca. Cuando por fin lo encuentra, escribe su nombre en pantalla y lo va incluyendo bloque por bloque. Todo esto ocurre en la primera pasada, puesto que el ensamblador necesita realizar las dos pasadas con el texto que se incluye.

El ejemplo del apéndice 3 utiliza este comando.

2.9 Macros

Una *macro*, forma abreviada de referirse a una macro-instrucción, es una rutina formada por yuxtaposición de operaciones elementales que utilizan argumentos ficticios como operandos. Una macro tiene la forma general siguiente:

```
Nombre:     MAC
            ⋮
            definición de la macro
            ⋮
            ENDM
```

'Nombre' es el nombre que se da a la macro y que servirá para invocarla (llamarla) cuando se desee utilizarla. Para hacer dicha llamada se debe escribir el nombre de la macro en el campo correspondiente a los códigos nemotécnicos. La definición de la macro consta de las líneas comprendidas entre las sentencias MAC y ENDM.

Una macro puede utilizar hasta 32 argumentos, que reciben el nombre de parámetros, y se numeran del Ø al 31. La definición de una macro describe los parámetros que utiliza mediante el signo ' = ' seguido del número de parámetro.

Por ejemplo, la macro

```
MOVER:      MAC
            LD   HL,=Ø
            LD   DE,=1
            LD   BC,=2
            LDIR
            ENDM
```

utiliza 3 parámetros, que carga en los pares de registros HL,DE y BC y con los que realiza la instrucción LDIR. Los parámetros representan pues, en este caso, la dirección de origen, la dirección de destino y el número de bytes que se trasladan mediante la operación LDIR. Cuando se desee utilizar la macro en un punto concreto, basta llamarla por su nombre seguido de los valores que se quiera dar a los parámetros; por ejemplo,

```
      MOVER 16384,16385,4Ø96
```

Se puede utilizar cualquier expresión válida para especificar el valor de un parámetro; por ejemplo,

```
      MOVER 16384,destino,long+1
```

Dentro de la definición de una macro, los parámetros pueden aparecer como términos de una expresión válida; es lo que ocurre en el ejemplo siguiente:

```
HMS:        MAC
            LD   HL,=Ø*36ØØ
            LD   DE,=1*6Ø
            ADD  HL,DE
            LD   DE,=2
            ADD  HL,DE
            ENDM
```

que es una macro con tres parámetros (horas, minutos y segundos) cuya función es acumular en el par HL el número total de segundos especificados por los parámetros. Un ejemplo de utilización de esta macro puede ser el siguiente:

```
Horas:      EQU  2
Minutos:    EQU  3Ø
Segundos:   EQU  12
Comienzo:   EQU  Ø

            HMS  Horas,Minutos,Segundos
            LD   DE,Comienzo
            ADD  HL,DE          ;HL acumula el tiempo final
```

Las macros no se pueden anidar entre sí. Dentro de la definición de una macro no se puede definir otra, ni tampoco se puede llamar una macro.

Cuando el ensamblador encuentra el nombre de una macro en el campo correspondiente a los códigos nemotécnicos, procede a ensamblar el texto de la macro. Generalmente el texto de la macro no figura en el listado de ensamblador, en el que aparece solamente el nombre de la macro. Pero se puede hacer también que suceda lo contrario. Los comandos '*M +' y '*M –' permiten elegir la opción que se desea en cada momento.

La utilización de macros requiere la reserva de un espacio en memoria para almacenar las definiciones durante el ensamblado; esto se hace, como veremos, con el comando 'K' del editor. La reserva de un espacio insuficiente genera un mensaje de error e impide la continuación del ensamblado.

2.10 GEN y el mapa de memoria de MSX

Como usted probablemente ya sabe, 32K de los 64K de RAM que posee un ordenador MSX de 64K no son accesibles para el MSX BASIC. El GEN puede perfectamente instalarse en esta zona, pero necesita utilizar parte de los 32K a los que la ROM puede acceder directamente para relacionarse con ella.

El principal área que se utiliza es el que corresponde al área de trabajo de la cola de sonido, entre las posiciones #F975 y #FA74.

El otro área que se utiliza es la comprendida entre #FFDC y #FFF9; no debe ser utilizada por ningún programa si se desea mantener GEN en el ordenador.

Cuando se ejecuta GEN, la ROM del BASIC no está presente aunque existe la posibilidad de llamarla. Para ello se hace una llamada a la rutina de la posición #F975, seguida de la dirección de la rutina de la ROM que se desea, de la que sólo debe darse el byte bajo. Puede emplearse una macro para realizar estas llamadas sistemáticamente. El ejemplo del apéndice 3 muestra cómo hacerlo.

Si desea que su programa pueda acceder permanente a la ROM, haga

```
CALL #FFE9
```

Para volver a la configuración de 64K debe hacerse la llamada

```
CALL #FFE2
```

Esto es válido para todos los ordenadores MSX, independientemente del tamaño de la RAM o del canal en que esté instalada.

El código que se carga utilizando el comando 'O' puede llamar a estas mismas rutinas para acceder a la ROM o cambiar la configuración de la RAM. Ahora bien, si su programa altera el área de la cola de sonido, no debe llamar a #F975. Cuando se utiliza el comando 'O' para mover código, éste se cargará en cualquier parte de la RAM no accesible a BASIC.

SECCIÓN 3. EL EDITOR

3.1 Introducción al editor

El editor que proporciona GEN es fundamentalmente un editor de línea, pero aprovecha al mismo tiempo las facilidades de edición de pantalla completa que poseen los ordenadores de tipo MSX.

Para reducir el tamaño del fichero de texto que se crea, el editor no guarda las líneas tal y como se escriben, sino que realiza previamente la eliminación de todos los espacios en blanco innecesarios. Mientras se teclea una línea, el editor almacena los caracteres que se escriben en una zona tampón de memoria que crea el propio programa. Cuando se pulsa ⟨RETURN⟩, la línea se transfiere al fichero de texto; durante esta transferencia se produce la reducción de espacios a que aludíamos. Se suprimen primero todos los espacios anteriores al primer caracter que no es un blanco; se comienza entonces la transferencia de caracteres hasta encontrar un blanco o ':'; entonces se inserta un tabulador y se eliminan los blancos que se encuentren hasta llegar a un carácter que no sea un blanco y así sucesivamente. El resultado es eliminar los espacios superfluos e introducir tabuladores entre los diversos campos de la línea (etiqueta, código nemotécnico, operandos y comentario). El proceso termina cuando se llega al carácter 'CR' producido por la tecla ⟨RETURN⟩.

El resultado es crear un fichero de texto cuyo almacenamiento es lo más económico posible y cuyo listado presentará gran claridad gracias a los tabuladores.

Lógicamente, se respetan los espacios en blanco que puedan aparecer en el comentario. Por otra parte, éste es el único campo en cuyo interior puede haber blancos, pues si se dejan blancos en el interior de la etiqueta, el código o los operandos, el editor separaría estos campos de forma errónea.

Cuando comienza la ejecución de GEN, se entra directamente en modo de edición. En la pantalla aparece el mensaje

    Copyright Hisoft 1984
    All rights reserved

seguido del cursor de edición. Se puede introducir entonces cualquiera de los comandos del editor; estos comandos tienen el formato general siguiente:

    C n1,n2,l1,l2   seguido de ⟨RETURN⟩

donde

C	es el nombre de uno de los comandos que describiremos a continuación;
n1 y n2	son enteros entre 1 y 65535;
l1 y l2	son cadenas literales con un máximo de 2Ø caracteres cada una;

los números n1 y n2 se pueden escribir en forma decimal, hexadecimal (con el signo '#' delante) o binaria (con el signo '%' delante). Los argumentos se separan con comas, aunque el símbolo separador se puede cambiar con el comando 'Q', como ya veremos. Los espacios en blanco se ignoran, salvo en lo que concierne a las cadenas literales.

Dependiendo del comando, los argumentos pueden ser obligatorios o no serlo. Algunos comandos sólo tienen efecto si se declaran explícitamente determinados argumentos. Otros no exigen la introducción explícita de los argumentos y, por defecto, toman los que se han utilizado en la última ocasión. De entrada, el editor tiene cargados como argumentos el número 1Ø para n1 y n2 y la cadena vacia para l1 y l2.

A cualquier error en la introducción de un comando, responde el editor con el mensaje 'Pardon?', y el comando queda anulado. Que l2 posea más de 2Ø caracteres es considerado un error; sin embargo, si l1 posee más de 2Ø caracteres, el editor se limita a ignorar los que sobran.

El nombre del comando es una letra, y es indiferente que se escriba mayúscula o minúscula.

Durante la introducción de una línea se pueden utilizar los comandos de edición habituales de MSX, como por ejemplo las teclas de movimiento del cursor, ⟨CTRL/F⟩, ⟨CTRL/E⟩, etc.

A continuación vamos a realizar una descripción de cada uno de los comandos del editor. El símbolo '⟨ ⟩' rodeando un argumento indica que el argumento es obligatorio para que el comando tenga efecto.

3.2 Comandos del editor

3.2.1 Creación e inserción de texto

Cuando se escribe una línea, ésta se inserta en el fichero de texto en el lugar que le corresponda, el cual viene determinado por el número de línea.

Para insertar líneas de texto basta escribir un número de línea, dejar un espacio y escribir el texto que se desee, terminando con ⟨RETURN⟩ para que la línea se archive. En cualquier momento se puede abandonar la escritura y volver al nivel de comandos del editor pulsando ⟨CTRL/STOP⟩.

Si se escribe un número de línea seguido de ⟨RETURN⟩ (es decir, una línea sin texto), entonces la línea con ese número es borrada del fichero de texto (si existía).

La tecla ⟨BS⟩ destruye el caracter anterior al cursor hasta llegar, si se desea, al comienzo de la línea, pero no más allá.

Como hemos dicho antes, el texto de la línea se va introduciendo en un tampón creado por el editor, en tanto no se pulse ⟨RETURN⟩. Si este tampón se llena, el resto de lo que usted escriba se ignora.

Si al escribir o insertar texto el editor detecta que llega al final de la memoria RAM, aparecerá el mensaje 'Bad Memory!' que indica que no cabe más texto en el fichero de texto. Entonces el fichero de texto debe guardarse en cinta total o parcialmente, para ser recuperado más adelante.

comando I **sintaxis: I n1,n2**
acción: insertar texto

El comando 'I' proporciona automáticamente números de línea, comenzando por el número n1 e incrementándolo cada vez en la cantidad n2. Cuando aparece en la pantalla el número de línea, se puede introducir el texto que se desee, terminando la línea con ⟨RETURN⟩, momento en el que aparecerá un nuevo número de línea. Para terminar se pulsa ⟨CTRL/STOP⟩.

Si se introduce así el número de una línea que ya existe, el editor da a la antigua un nuevo número de línea. Si lo que se desea introducir es un bloque de líneas, convendrá entonces utilizar el valor n2 = 1 para asegurarse de que la inserción es correcta.

Si el incremento automático del número de línea lleva a un número de línea superior a 65535, entonces se termina automáticamente el modo 'inserción' y se pasa al nivel de comandos del editor.

3.2.2 Listado de texto

Para ver el contenido del fichero de texto se pueden utilizar los comandos siguientes, que proporcionan un listado del mismo en la pantalla o en la impresora. En el listado, las líneas aparecen tabuladas, separándose claramente los distintos campos de la línea.

comando L **sintaxis:** **L n1,n2**
acción: listado en pantalla

Este comando tiene por efecto listar en la pantalla las líneas comprendidas entre la n1 y la n2 inclusive. Los valores por defecto de los argumentos son 1 y 65535, respectivamente (no son, pues, los anteriormente usados). Si se utiliza 'L' sin argumentos, se obtiene el listado completo.

El número de líneas que aparecen simultáneamente en la pantalla es de 22; cuando ya hay 22 líneas en pantalla, el listado se detiene. Si aún no se ha llegado a la línea n2, se obtienen otras 22 líneas pulsando cualquier tecla normal. Si se pulsa ⟨CTRL/STOP⟩ se vuelve al nivel de comandos del editor.

comando Z **sintaxis:** **Z n1,n2**
acción: listado por impresora

Es similar al anterior, pero lanza el listado por la impresora. Si no hay impresora conectada, o si la impresora está desactivada (''off-line''), aparece en pantalla el aviso 'No Printer!' y el comando no tiene efecto. Las teclas ⟨CTRL/STOP⟩ interrumpen el listado y hacen que se vuelva el nivel de comandos.

comando Y **sintaxis:** **Y n1**
acción: tamaño de la página del listado

Permite fijar en n1 el número de líneas por página en los listados de ensamblador que salen por la impresora. Por ejemplo, 'Y 6Ø ⟨RETURN⟩' hace que el listado tenga 6Ø líneas por página, es decir, que cada 6Ø líneas se genere un salto automático de página.

3.2.3 Edición de texto

La mayor parte de las veces es inevitable tener que modificar algunas líneas de texto. Existen varios comandos que permiten corregir, borrar, alterar y renumerar las líneas.

comando E **sintaxis:** **E n**
acción: edita la línea número n

Edita la línea número n para permitir su modificación. Si la línea n no existe, el comando queda sin efecto. Si existe, la línea se copia en un tampón de la memoria y además aparece en pantalla (número de línea incluido). La línea se puede corregir entonces con ayuda de las flechas de movimiento del cursor y de los comandos de edición del MSX, que son los siguientes:

⟨CTRL/B⟩	mueve el cursor a la palabra anterior,
⟨CTRL/C⟩	vuelve al modo comando,
⟨CTRL/E⟩	borra hasta el final de la línea,
⟨CTRL/F⟩	lleva el cursor al comienzo de la siguiente palabra,
⟨BS⟩	borra el carácter que precede al cursor,
⟨HOME⟩	lleva el cursor a la esquina superior izquierda de la pantalla,
⟨SHIFT/HOME⟩	borra completamente la pantalla,
⟨RETURN⟩	valida la línea,
⟨CTRL/N⟩	lleva el cursor al final de la línea,
⟨INS⟩	bascula entre los modos de inserción y sobreescritura
⟨CTRL/U⟩	borra la línea,
⟨DELETE⟩	borra el carácter sobre el que está el cursor.

Cuando se han concluido las modificaciones, se pulsa ⟨RETURN⟩ para validar la línea e insertarla en el fichero de texto.

comando D **sintaxis:** **D ⟨n1,n2⟩**
acción: borra líneas

Sirve para suprimir del fichero de texto todas las líneas comprendidas entre la n1 y la n2 inclusive. Los argumentos son obligatorios; la ausencia de alguno de ellos o el hecho de que n2 sea menor que n1 dejan sin efecto el comando. Esto se hace para prevenir las catástrofes que pudiera originar algún descuido. Para borrar solamente la línea n, basta hacer n1 y n2 iguales a n, aunque se logra lo mismo escribiendo el número n seguido de ⟨RETURN⟩.

comando M **sintaxis:** **M ⟨n1,n2,n3⟩**
acción: mueve un bloque de líneas

Hace que el bloque de las líneas que van de la n1 a la n2 inclusive se coloque inmediatamente antes de la línea n3, para lo que se renumeran las líneas n3 y posteriores si es necesario. El bloque original se borra. Los tres argumentos son obligatorios.

No existe límite para el tamaño del bloque que se mueve, puesto que las líneas se mueven de una en una.

comando N **sintaxis:** **N ⟨n1,n2⟩**
acción: renumera las líneas

Renumera el fichero de texto comenzando por el número n1 y dando cada vez un incremento de n2 en el número de línea. Los dos argumentos son obligatorios. Si la renumeración hace que algún número de línea exceda de 65535, entonces se conserva la numeración original.

comando F **sintaxis:** **F n1,n2,l1,l2**
comando S **sintaxis:** **S**
acción: búsqueda y sustitución

Los comandos 'F' y 'S' de búsqueda ("find") y sustitución se usan normalmente combinados. El comando 'F' busca la cadena literal l1 a lo largo de las líneas comprendidas entre la n1 y la n2, recorriendo estas líneas en el sentido de avance del texto. Cuando la encuentra, procede a editar la línea en que aparece la cadena; se puede actuar entonces como en el caso del comando 'E'. Pero también se puede pulsar la letra 'S', y en tal caso la cadena l1 queda sustituida por la cadena l2. Además, la sustitución produce una nueva búsqueda de la cadena l1. Pulsando repetidamente 'S' y ⟨RETURN⟩ se realiza entonces la búsqueda y sustitución repetida con bastante rapidez.

Cuando no se especifican argumentos en el comando 'F', se toman los empleados en la última ocasión. Esto permite que, para buscar repetidamente la misma cadena, baste con introducir el comando sin parámetros a partir de la segunda vez.

3.2.4 Grabación y carga de ficheros

Los comandos 'P' y 'G' sirven para grabar en cinta o leer de la cinta un fichero de texto. El comando 'V' permite comprobar la concordancia entre un fichero de texto y un fichero almacenado en cinta. El comando 'O' sirve para grabar en cinta código objeto.

comando P **sintaxis:** **P n1,n2,l1**
acción: graba en cinta un fichero de texto

Graba en cinta el fichero formado por las líneas comprendidas entre la n1 y la n2 inclusive, con el nombre dado por la cadena literal l1. Recuerde que se usarán los argumentos anteriormente empleados si no se proporcionan explícitamente otros. El mensaje

'Press REC/PLAY then any key'

le recordará que debe poner el magnetófono en posición de grabar y pulsar después una tecla cualquiera.

La duración del tono de cabecera del fichero grabado, que por defecto es de 1 segundo aproximadamente, se puede alargar hasta unos 5 segundos si se pone el símbolo '!' como primer carácter del nombre del fichero. Esto suele ser aconsejable cuando posteriormente se va a utilizar este fichero para incluirlo en el texto en memoria.

comando G **sintaxis:** **G ,,l1**
acción: carga de cinta un fichero de texto

Carga el fichero que haya en la cinta con el nombre dado por la cadena l1 y lo incluye al final del texto que pueda haber en la memoria. Cuando en la memoria exista efectivamente texto, el nuevo texto completo será renumerado con números de línea que comienzan por el 1 y aumentan de 1 en 1. El mensaje

'Press PLAY then any key'

le recordará que debe poner el magnetófono en posición de lectura.

Cuando el primer carácter del argumento l1 es '!', se ignora este carácter.

comando V **sintaxis:** **V ,,l1**
acción: verificación de un fichero de texto

Verifica la concordancia entre el fichero de texto de la memoria y el fichero que haya en la cinta con el nombre dado por la cadena l1. Cuando el primer carácter en l1 es '!', se ignora este carácter.

Aparecerá el mensaje 'Verified' o el mensaje 'Failed!' (''fallado'') según que los dos ficheros posean o no el mismo texto.

comando O **sintaxis:** **O ,,l1**
acción: graba en cinta código objeto

Supongamos que se desea grabar en cinta o cargar desde la cinta el código objeto de un programa (que es un fichero binario). Si el emplazamiento del programa en la memoria es por encima de la dirección #8ØØØ, lo más cómodo es volver al BASIC con el comando 'B' y utilizar entonces los comandos BSAVE y BLOAD. Sin embargo, esto no es posible cuando el programa va en la parte de la RAM no accesible a BASIC.

El comando 'O' sirve para resolver este problema. Lo que hace es grabar antes del programa un pequeño programa cargador colocado en la zona #F975 a #FAF4 (área de la cola de sonido); este programa cargador se encargará justamente de cargar el código objeto y comenzar su ejecución en la dirección señalada con la seudo-operación END. El programa cargador pasará a la cinta con el nombre dado por la cadena l1.

Para realizar después el proceso inverso, o sea, la carga del programa en memoria, se debe utilizar el comando BLOAD con la opción ',R'. Por ejemplo, si para la grabación se ha utilizado el comando

O ,,PRUEBA

ahora habrá que cargarlo desde BASIC con el comando

BLOAD ''PRUEBA'',R

que tendrá por efecto cargar el programa y comenzar su ejecución.

Si en su programa ha utilizado varias veces la seudo-operación ORG, el comando 'O' sólo grabará el código que exista entre la dirección señalada por el último ORG y el final del programa.

Conviene leer también lo que explicamos en 2.1Ø en relación con este comando.

3.2.5 Ensamblado y ejecución

comando A **sintaxis:** **A**
acción: llama al ensamblador

Llama al ensamblador, que actúa como ya se explicó en la sección 2; al terminar el proceso de ensamblado se vuelve al editor.

comando R **sintaxis:** **R**
acción: ejecuta el programa

Ejecuta el programa objeto, siempre que no haya habido errores durante el ensamblado y que la seudooperación ENT haya señalado la posición de arranque de la ejecución. Si en el programa aparece la instrucción RET (de código #C9), el control volverá al editor, siempre que la pila continúe igual que al comienzo de la ejecución.

Recuerde que ENT habrá quedado sin efecto si el ensamblado se ha realizado con la opción 16.

3.2.6 Otros comandos

comando H **sintaxis:** **H**
acción: pantalla de ayuda

Muestra la pantalla de ayuda, que recuerda cuáles son los comandos del editor.

comando B **sintaxis:** **B**
acción: pasa a BASIC

Devuelve el control al sistema operativo. Para volver a GEN desde BASIC utilice lo siguiente (arranque en caliente):

DEFUSR=&HFFFØ ⟨RETURN⟩
A=USR(Ø) ⟨RETURN⟩

comando Q **sintaxis:** **Q ,,l1**
acción: cambio de separador de los argumentos

De entrada, el símbolo que delimita los diferentes argumentos de un comando es la coma. El comando 'Q' cambia este delimitador por el primer carácter de la cadena literal l1; este carácter no puede ser un espacio. Una vez que el delimitador se ha cambiado, se debe utilizar siempre el nuevo, incluso en el propio comando 'Q'. El comando 'C' le dirá cuál es el actual delimitador si se le ha olvidado.

comando C **sintaxis:** **C**
acción: informa sobre los argumentos

Muestra cuáles son los valores actuales del delimitador y de los argumentos n1, n2, l1 y l2. Si va a utilizar en un comando los argumentos por defecto, puede ser interesante comprobar previamente si son los que desea.

comando X **sintaxis:** **X**
acción: comienzo y final del fichero de texto

Muestra cuáles son las posiciones (en notación decimal) del comienzo y del final del fichero de texto. Mediante este comando se puede saber cuánta memoria queda después del fichero de texto. La posición del final del fichero de texto se almacena en la dirección 'comienzo de GEN + 28'. Conociendo estas direcciones, se puede "hurgar" en el fichero de texto desde BASIC o desde MON.

comando K **sintaxis: K n**
acción: tamaño del tampón de las macro

Reserva una zona tampón de n bytes para almacenar las definiciones de las macro-instrucciones en tanto dura el ensamblado. Por defecto, se reserva una zona de 1ØØ bytes. El mensaje 'No Macro Space!' avisa de que el tamaño reservado es insuficiente; en tal caso, se lo debe aumentar con este comando.

comando J **sintaxis: J**
acción: definible a voluntad del usuario

Permite al usuario de GEN crear sus propios comandos del editor. El comando 'J' realiza una llamada a la posición de memoria 'comienzo de GEN + 3'. En dicha posición lo que se encuentra es el código #C3 de la instrucción 'JP' del Z8Ø, por lo que, si usted introduce en 'comienzo de GEN + 4 y + 5' la dirección de alguna rutina que le interese, podrá llamarla desde el editor con el comando 'J'.

comando U **sintaxis: U**
acción: último número de línea

Muestra el número de línea de la última línea del fichero de texto. Es útil para añadir texto, listar las últimas líneas, etc.

comando W **sintaxis: W n1,n2**
acción: volcado de memoria

Proporciona un listado en pantalla del contenido de la memoria entre las posiciones n1 y n2. El listado se detiene pulsando 〈CTRL/STOP〉. Si a continuación se vuelve a pulsar 〈CTRL/STOP〉, el control vuelve al editor; si se pulsa otra tecla, se reanuda el listado.

3.3 Un ejemplo de utilización del editor

Supongamos que, con el comando 'I 1Ø,1Ø', escribe usted el siguiente programa:

```
 1Ø *H      NUMEROS ALEATORIOS DE 16 BITS
 2Ø
 3Ø ;ENTRADA: HL con un num. semilla para generar un num. aleatorio
 4Ø ;SALIDA:  HL con el nuevo numero aleatorio
 5Ø
 6Ø Random: PUSH AF      ;guardar los registros
 7Ø         PUSH BC
 8Ø         PUSH HL
 9Ø         ADD HL,HL    ;*2
1ØØ         ADD HL,HL    ;*4
11Ø         ADD HL,HL    ;*8
12Ø         ADD HL,HL    ;*16
13Ø         ADD HL,HL    ;*32
14Ø         ADD HL,HL    ;*64
15Ø         PIP BC       ;numero anterior
16Ø         ADD HL,DE
17Ø         LD  DE,41
18Ø         ADD HL,DE
19Ø         POP BC       ;restaurar los registros
2ØØ         POP AF
21Ø         REY
```

Leyéndolo más despacio notará que se han introducido algunos errores:

- lin 1Ø: la 'h' del comando '*H' del ensamblador debe ser mayúscula;
- lin 4Ø: se ha escrito 'aleatotio' en lugar de 'aleatorio';
- lin 15Ø: se ha escrito 'PIP' en lugar de 'POP';
- lin 16Ø: conviene añadir un comentario;
- lin 21Ø: se ha escrito 'REY' en lugar de 'RET'.

Además, hay que añadir aún dos líneas con 'ADD HL,HL' entre la 14Ø y la 15Ø, y hay que sustituir el par DE por el par BC en las líneas 16Ø a 18Ø. Vamos a describir la secuencia de comandos y operaciones que permite hacer las oportunas rectificaciones.

—Pulsar 'E 1Ø ⟨RETURN⟩', mover el cursor hasta la letra h y pulsar 'H' y ⟨RETURN⟩.
—Pulsar 'F 4Ø,4Ø,aleatotio,aleatorio ⟨RETURN⟩', pulsar ⟨RETURN⟩ y luego 'S ⟨RETURN⟩'.
—Pulsar 'I 142,2 ⟨RETURN⟩' y escribir las líneas

```
142 ADD HL,HL   ;*128 ⟨RETURN⟩
144 ADD HL,HL   ;*256 ⟨RETURN⟩
```

cuando aparezcan los números 142 y 144; luego, pulsar ⟨CTRL/STOP⟩.
—Pulsar 'F 15Ø,15Ø,PIP,POP ⟨RETURN⟩', pulsar ⟨RETURN⟩ y luego 'S ⟨RETURN⟩'.
—Pulsar 'E 16Ø ⟨RETURN⟩', ir hasta el final de la línea y escribir ';*257+41 ⟨RETURN⟩'.
—Pulsar 'F 16Ø,18Ø,DE,BC ⟨RETURN⟩' y utilizar repetidamente el comando 'S'.
—Pulsar 'E 12Ø ⟨RETURN⟩', mover el cursor hasta la letra Y y pulsar 'T' y ⟨RETURN⟩.
—Pulsar 'N 1Ø,1Ø ⟨RETURN⟩' para renumerar el fichero.

Es muy conveniente que practique usted con el ejemplo que acabamos de exponer, utilizando realmente el editor.

APÉNDICE 1. MENSAJES DE ERROR

ERROR 1	Error en el contexto de esta línea.
ERROR 2	Código nemotécnico no reconocido.
ERROR 3	Instrucción mal escrita.
ERROR 4	Se asigna valor a un símbolo más de una vez.
ERROR 5	Esta línea contiene un carácter ilegal, no válido en el contexto en el que aparece.
ERROR 6	Uno de los operandos de esta línea es ilegal.
ERROR 7	Se emplea como símbolo una palabra reservada.
ERROR 8	Confusión de los registros.
ERROR 9	Demasiados registros en la línea.
ERROR 1Ø	El valor de una expresión que no debe sobrepasar los 8 bits excede de dicha longitud.
ERROR 11	Las instrucciones JP (IX+n) y JP (IY+n) son ilegales.
ERROR 12	Error en la sintaxis de una seudo-operación.
ERROR 13	EQU asigna a una etiqueta un valor que no ha sido definido.
ERROR 14	División por Ø.
ERROR 15	El resultado de una multiplicación sobrepasa 32767 (''overflow'').
ERROR 16	Es ilegal la anidación de una macro en otra.
ERROR 17	Este nombre no corresponde a ninguna macro.
ERROR 18	Es ilegal llamar a una macro dentro de otra.
ERROR 19	Es ilegal la anidación de sentencias condicionales.
Bad ORG!	Una sentencia ORG se refiere a una dirección con la que el código objeto se escribiría sobre GEN, sobre el fichero de texto o la tabla de símbolos. (El control pasa al editor.)
Out of Table space!	No se ha reservado suficiente memoria para la tabla de símbolos. (El control pasa al editor.)
Bad Memory!	El fichero de texto ocupa ya prácticamente hasta el final de la RAM y no puede añadirse más texto. (Hay que grabar en cinta todo el texto o parte de él.)
No Macro Space!	No hay espacio suficiente para almacenar las definiciones de las macros. (Se debe asignar más memoria con el correspondiente comando del editor.)

APÉNDICE 2. PALABRAS RESERVADAS, CÓDIGOS, ETC.

Lo que sigue es una lista de las palabras reservadas de GEN. Estas palabras no pueden ser utilizadas como etiquetas, aunque si pueden formar parte de una etiqueta. Nótese que se trata en todos los casos de palabras compuestas por letras mayúsculas.

```
A    B    C    D    E    H    L    I    R    $

AF        AF        BC        DE        HL        IX

IY        SP        NC        Z         NZ        M

P         PE        PO
```

Esta otra lista contiene los códigos nemotécnicos de las operaciones del Z8Ø, de las seudo-operaciones del ensamblador y de los comandos del ensamblador. No son palabras reservadas. Nótese que se trata en todos los casos de palabras compuestas por letras mayúsculas.

```
ADC   ADD   AND   BIT   CALL  CCF   CP    CPD   CPDR

CPI   CPIR  CPL   DAA   DEC   DI    DJNZ  EI    EX

EXX   HALT  IM    IN    INC   IND   INDR  INI   INIR

JP    JR    LD    LDD   LDDR  LDI   LDIR  NEG   NOP

OR    OTDR  OTIR  OUT   OUTD  OUTI  POP   PUSH  RES

RET   RETI  RETN  RL    RLA   RLC   RLCA  RLD   RR

RRA   RRC   RRCA  RRD   RST   SBC   SCF   SET   SLA

SRA   SRL   SUB   XOR


DEFB  DEFM  DEFS  DEFW  ELSE  END   ENDM  ENT   EQU

IF    MAC   ORG


*E    *H    *L    *S    *C    *M    *F
```

APÉNDICE 3. UN EJEMPLO DESARROLLADO

Lo que presentamos ahora es el ejemplo de una típica sesión de trabajo con el ensamblador. Si nunca ha trabajado con un ensamblador, convendrá que desarrolle este ejemplo antes de pasar al estudio de las secciones anteriores; esto le facilitará la comprensión de las mismas.

El símbolo ⟨RETURN⟩ no aparece realmente en los listados; lo incluimos para recordarle que debe usted pulsar la tecla correspondiente. Lo mismo sucede con ⟨CTRL/STOP⟩.

Objetivo de la sesión:

Escribir y comprobar una rutina rápida para multiplicar números enteros. Grabar en cinta el texto de la rutina, utilizando el comando 'P' del editor, para poder incluirla en futuros programas.

Plan de trabajo:

1. Escribir la rutina y grabarla en cinta, sin preocuparse por el momento de que pueda tener errores.
2. Depurar la rutina, utilizando los comandos de edición.
3. Crear un programa que llame a la rutina de multiplicación para comprobar su funcionamiento.
4. Grabar la rutina en cinta con el comando 'P', para incluirla en otros programas.

Comience por cargar GEN. Para ello utilice desde BASIC el comando

RUN "CAS:GENMSX" ⟨RETURN⟩

y ponga en marcha la cinta del magnetófono. Cuando GEN se haya cargado, aparecerá la pantalla de ayudas seguida de un cursor; está usted en el editor (en modo de edición), donde puede crear programas.

Fase 1. Escritura de la rutina

Teclee en primer lugar

I 10,10 ⟨RETURN⟩

y a continuación todo lo que sigue, a medida que vayan apareciendo los números de línea.

```
 10 ;Una rutina rapida para multiplicar.
 20 ;Multiplica HL por DE y entrega el
 30 ;resultado en HL. El indicador C
 40 ;se arrastre se activa en caso de
 50 ;sobrepasamiento <RETURN>
 60 <RETURN>
 70         ORG  #D800 <RETURN>
 80 <RETURN>
 90 Mult:   OR <RETURN>
100         SBC  HL,DE ;HL>DE? <RETURN>
110         ADD  HL,DE <RETURN>
120         JR   NC,Mul ;si <RETURN>
130         EX   DE,HL <RETURN>
140 Mul:    OR   D <RETURN>
```

```
15Ø          SCF  ;comprobar sobrepasamiento <RETURN>
16Ø          RET  NZ ;DE>255 <RETURN>
17Ø          OR   E :por Ø? <RETURN>
18Ø          LD   E,D <RETURN>
19Ø          JR   NZ,MU4 ;no <RETURN>
2ØØ          EX   DE,HL ;Ø <RETURN>
21Ø          RET  <RETURN>
22Ø <RETURN>
23Ø ;rutina principal. <RETURN>
24Ø <RETURN>
25Ø Mu2:     EX   DE,HL <RETURN>
26Ø          ADD  HL,DE <RETURN>
27Ø          EX   DE,HL <RETURN>
28Ø Mu3:     ADD  HL,HL <RETURN>
29Ø          RET  C ;sobrepasamiento <RETURN>
3ØØ Mu4:     RRA  <RETURN>
31Ø          JR   NC,Mu3 <RETURN>
32Ø          OR   A <RETURN>
33Ø          JR   NZ,Mu2 <RETURN>
34Ø          ADD  HL,DE <RETURN>
35Ø          RET  <RETURN>
36Ø <CTRL/STOP>
```

Ahora teclee

P 1Ø,35Ø,Mult ⟨RETURN⟩

para grabar la rutina.

Ha utilizado el comando 'I' para insertar texto y el comando 'P' para grabar la rutina en cinta con el nombre de 'Mult'. Habrá notado que el ordenador le ha recordado la necesidad de poner el magnetófono en posición de grabación cuando era necesario.

Fase 2. Depuración de la rutina

En primer lugar comprobaremos si el texto se ensambla correctamente. Utilizaremos la opción 6 del ensamblador para no generar el código objeto ni ningún listado por el momento. Llame al ensamblador con

A ⟨RETURN⟩
Table size: ⟨RETURN⟩ (el tamaño por defecto)
Options:6 ⟨RETURN⟩

y obtendrá en la pantalla:

```
Hisoft GEN Assembler           Page   1

Pass 1 errors: ØØ


Pass 2 errors: ØØ

*WARNING* MU4  absent
Table used:      76  from    16Ø
```

El ensablador nos avisa de que hemos intentado saltar a la etiqueta 'MU4' que no estaba definida; en realidad lo que se quería era saltar a 'Mu4' en la línea 190. Hay que editar esta línea y corregirla; pulse

F190,190,MU4,Mu4<RETURN>

y verá aparecer

```
190          JR  NZ,MU4   ;no
```

Pulse <RETURN> y luego 'S <RETURN>' para realizar la sustitución. Proceda a ensamblar de nuevo la rutina y comprobará que se realiza sin errores.

FASE 3. PROGRAMA QUE UTILIZA LA RUTINA

Sin eliminar nuestra rutina de la memoria, vamos a crear un programa que sirva para probarla. Teclee

I 1,1 <RETURN>

y, a medida que van apareciendo números de línea, teclee

```
 1 ;Una macro para llamar la ROM
 2 ROM:    MAC <RETURN>
 3         CALL #F975 <RETURN>
 4         DEFW =0 <RETURN>
 5         ENDM <RETURN>
 6 <RETURN>
 7 ;Un programa para probar <RETURN>
 8 ;la rutina Mult. <RETURN>
 9         ORG  #D900 <RETURN>
10         LD   HL,50 <RETURN>
11         LD   DE,20 <RETURN>
12         CALL Mult ;Multiplicar <RETURN>
13         LD   A,H ;mostrar el resultado <RETURN>
14         CALL SalA <RETURN>
15         LD   A,K <RETURN>
16         CALL SalA <RETURN>
17         LD   A,13
18         ROM #A2
19         LD   A,10
20         ROM #A2
21         RET ;Vuelta al editor <RETURN>
22 <RETURN>
23 ;rutina para mostrar A en hex <RETURN>
24 <RETURN>
25 SalA:   PUSH AF <RETURN>
26         RRCA <RETURN>
27         RRCA <RETURN>
28         RRCA <RETURN>
29         RRCA <RETURN>
30         CALL Bin4 <RETURN>
31         POP  AF <RETURN>
32 Bin4:   AND  %1111 <RETURN>
33         ADD  A,#90 <RETURN>
34         DAA <RETURN>
35         ADC  A,#40 <RETURN>
36         DAA <RETURN>
37         ROM #A2 ;ROM envia caracter a la pantalla <RETURN>
38         RET <RETURN>
39 <CTRL/STOP>
```

A continuación ensamble el fichero de texto, que contendrá tanto el programa como la rutina Mult; pulse

A ⟨RETURN⟩
Table size: ⟨RETURN⟩
Options:6 ⟨RETURN⟩

y obtendrá en la pantalla

```
Hisoft GEN Assembler              Page    1

Pass 1 errors: ØØ

D9ØD   3EØD          15                  LD    A,K
*ERROR* 1Ø   [pulse cualquier tecla para continuar]

Pass 2 errors: Ø1

*WARNING* K absent
Table used:      1Ø8  from     21Ø
```

Se ha deslizado un error en la línea 15; no debe ser 'LD A,K', sino 'LD A,L'. Utilice el comando 'E':

E15 ⟨RETURN⟩

y obtendrá

15   LD A,K

Ahora lleve el cursor hasta la letra K y pulse 'L ⟨RETURN⟩'.

Si todo se ha hecho correctamente no habrá nuevos errores de ensamblado. Llame al ensamblador y utilice la opción 4: comprobará en principio que todo es correcto (supondremos que así sucede realmente).

Estamos en condiciones de comprobar el funcionamiento de la rutina. Para ello debemos decir al editor dónde queremos que comience la ejecución, lo que haremos con la seudo-operación ENT. Como hay que introducir una línea intermedia, empezaremos por renumerar el fichero de texto. Teclee

N1Ø,1Ø ⟨RETURN⟩
95 ENT $ ⟨RETURN⟩

Ahora ensamble otra vez el texto. El ensamblado terminará correctamente con los mensajes

```
Table used:      1ØØ  from     2Ø8
Executes: XXXXX
```

o algo parecido. Podemos ya ejecutar el código objeto mediante el comando 'R', para lo que hay que pulsar

R ⟨RETURN⟩

Como estamos multiplicando 5Ø por 2Ø debería darnos 1ØØØ, que es #3E8 en hexadecimal. Sin embargo, lo que se obtiene es

ØØ32

¡No funciona! ¿Por qué?

Liste las líneas 47Ø a 59Ø (con 'L 47Ø,59Ø ⟨RETURN⟩') para que pueda leerlas. Observe en la línea 52Ø la instrucción 'OR D' seguida de 'SCF' y de un 'RET NZ'. Lo que pretende este conjunto de ins-

trucciones es asegurarse de que el resultado de la multiplicación no sobrepasará los 2 bytes. Para ello comprueba que en DE no hay un valor mayor que 255, o sea, que en D hay un Ø; si la comprobación es negativa queda activado el indicador de arrastre. Para comprobar que en el registro D hay un Ø se realiza la operación lógica 'A OR X'; esto proporciona el resultado apetecido cuando en el acumulador hay un Ø previamente, lo que no está garantizado. Observando ahora las líneas precedentes vemos que la instrucción 'OR A' de la línea 47Ø se puede sustituir por 'XOR A' que, además de disponer adecuadamente los indicadores para 'SBC HL,DE', introduce un Ø en A.

Haga, pues, el cambio con el comando 'E47Ø⟨RETURN⟩'; cuando aparezca

```
47Ø Mult:        OR A
```

lleve el cursor hasta ponerlo sobre la O y pulse '⟨INS⟩ X ⟨RETURN⟩'.

A continuación tendrá que ensamblar de nuevo (utilice la opción 4) y ejecutar el programa con el comando 'R'. Obtendrá ahora la respuesta correcta: #3E8.

Editando las líneas 1ØØ y 11Ø del programa, cambiando en ellas los números 5Ø y 2Ø por otros, ensamblando y ejecutando de nuevo, comprobará cuantas veces quiera que la rutina trabaja ahora perfectamente.

FASE 4. GRABACIÓN EN CINTA DE LA RUTINA

Puesto que hemos perfeccionado la rutina, convendrá guardarla en cinta en su nueva versión para utilizarla cuando la necesitemos. Para ello teclee

```
P 39Ø,999,Mult ⟨RETURN⟩
```

Si se conserva grabada la rutina, se la podrá incluir en los programas que se desee. La forma de hacerlo es la siguiente

```
5ØØ          RET
51Ø
52Ø ;Aqui se incluye la rutina Mult
53Ø
54Ø *F Mult
55Ø
56Ø ;siguiente rutina
```

Cuando el ensamblador llegue a la línea 54Ø se ocupará de leer la rutina Mult, lo que exige tener la cinta preparada para su lectura. Esto será así tanto en la primera pasada como en la segunda, por lo que puede ser cómodo disponer de dos grabaciones consecutivas de la rutina para no tener que rebobinar la cinta.

Estudie detenidamente el ejemplo y, sobre todo, hágalo usted mismo.

SEGUNDA PARTE

MON

SECCIÓN 1. INTRODUCCIÓN

Es usted el afortunado propietario (y esperamos que esto equivalga a decir comprador) de un programa desensamblador/depurador sencillo y potente, que puede ser usado en cualquier ordenador MSX que tenga al menos 16K de memoria.

Para lo que sirve MON es para ayudarle a depurar sus programas en código de máquina, tanto si han sido escritos directamente o creados con un ensamblador (posiblemente GEN) o desde BASIC.

1.1 Carga del programa

Para cargar MON, coloque la cinta en el magnetófono de manera que quede hacia arriba la cara marcada con 'MON debugger'; teclee

RUN "CAS:MONMSX" ⟨RETURN⟩

y ponga el magnetófono en posición de lectura. MON se cargará y comenzará automáticamente a ejecutarse.

En la pantalla aparecerá una buena cantidad de números, letras y símbolos cuyo significado vamos a explicarle. En concreto, en la parte superior izquierda de la pantalla, verá el símbolo '⟩' que le indica que puede introducir el comando que desee.

1.2 Lo que aparece en su pantalla

Lo que hace MON es convertir la pantalla en una especie de *tablero de control*, repleto de informaciones que le indican el estado de su ordenador.

La información que le proporciona esta pantalla transformada en tablero de control se divide en tres partes.

Estado de los registros

La parte que tiene el aspecto

```
                      (seudo-registro de canal y contador de programa)
 ØØ     PC ØØØØ       (puntero de pila)
        SP DB85       (registro índice IX)
        IX ØØØØ       (registro índice IY)
        IY ØØØØ       (registros HL' y HL)
ØØØØ    HL ØØØØ       (registros DE' y DE)
ØØØØ    DE ØØØØ       (registros BC' y BC)
ØØØØ    BC ØØØØ       (registros AF' y AF)
ØØØØ    AF ØØFF       (seudo-registro de memoria)
       .MR ØØØØ       (reg. de interrupción y reg. de regeneración de la memoria)
        IR ØØ3D       (estatus de interrupción y de los indicadores)
 I SZ H VNC
```

consiste en una exposición del contenido de los registros del Z8Ø (incluidos los registros alternativos), con su habitual ordenación en pares capaces de funcionar como registros de 16 bits.

Muestra también el contenido de dos seudo-registros (que no son registros del Z8Ø, sino creación de MON), el de canal ("slot") y el de memoria. El registro de canal describe cómo está configurado su ordenador en ese momento. El registro de memoria (simbolizado por MR) es útil para el almacenamiento de direcciones de memoria a las que habrá que volver más adelante; funciona de manera similar a la de la memoria de las calculadoras de bolsillo.

Se exhibe también el estado de la habilitación de interrupciones y el estado de cada uno de los indicadores ("flags"); el estado es 1 ("on" o "set") si aparecen escritos y Ø ("off" o "reset") si su posición aparece en blanco. En la figura anterior están todos a 1. Mientras se ejecuta, MON mantiene inhibidas las interrupciones.

Notará que aparece un punto '.' delante del registro MR. Diremos que MR es el 'registro seleccionado' en este momento. El punto de selección puede cambiar de un registro a otro, cambiando así el registro seleccionado; describiremos más adelante el comando que permite hacerlo.

Desensamblado de la memoria

Aparece en el panel que se encuentra a la izquierda del anterior. Tiene el aspecto

```
ØØØØ DI
ØØØ1 JP   #Ø2D7
ØØØ4 CP   A
ØØØ5 DEC  DE
ØØØ6 SBC  A,B
ØØØ7 SBC  A,B
ØØØ8 JP   #2683
ØØØB NOP
ØØØC JP   #Ø1B6
ØØØF NOP
ØØ1Ø JP   #2686
```

y consiste en el desensamblado de una porción de memoria que abarca 11 instrucciones (que es lo que cabe en el espacio de pantalla que tiene reservado). Inicialmente comienza en la dirección Ø de la memoria, pero puede hacerse que comience en la dirección contenida en el contador de programa (PC) utilizando el comando 'L'. Cuando el registro PC apunta a la dirección de alguna de las sentencias desensambladas, aparece el símbolo '}' a la derecha de dicha sentencia.

Contenido de la memoria

Debajo del anterior, aparece otro panel cuyo aspecto es el siguiente:

```
FFEØ ØØ ØØ FF FF ØØ ØØ FF FF ........
FFE8 ØØ ØØ FF FF ØØ ØØ FF FF ........
FFFØ ØØ ØØ FF FF ØØ ØØ FF FF ........
FFF8 ØØ ØØ FF FF ØØ ØØ FF FØ .......p
ØØØØ>F3<C3 D7 Ø2 BF 1B 98 98 sCW.?...
ØØØ8 C3 83 26 ØØ C3 B6 Ø1 ØØ C.&.C6..
ØØ1Ø C3 86 26 ØØ C3 D1 Ø1 ØØ C.&.CQ..
ØØ18 C3 45 1B ØØ C3 17 Ø2 ØØ CE..C...
ØØ2Ø C3 6A 14 ØØ C3 5E Ø2 ØØ Cj..C^..
ØØ28 C3 89 26 A1 13 ØØ ØØ ØØ C.&!....
```

Se trata de la fotografía de un sector de 80 bytes de la memoria centrado inicialmente alrededor de la posición 0. A la izquierda figuran direcciones de memoria (en hexadecimal). A la derecha de cada dirección figura el contenido de dicha posición de memoria y de las 7 siguientes (en hexadecimal). Más a la derecha aparece finalmente la representación ASCII del contenido de los 8 bytes de la línea, con el signo '.' cuando no es útil interpretar el correspondiente código como un carácter.

Observe que uno de los bytes está rodeado del corchete '〉 〈'; es en este caso el que corresponde a la posición 0 de memoria. Llamaremos 'puntero de memoria' a la dirección a la que señala dicho corchete; es la dirección en la que se centra el panel de contenido de la memoria. El valor del puntero de memoria puede seleccionarse de las dos maneras que vamos a ver.

Por defecto, el valor del puntero de memoria coincide con el contenido del registro que está seleccionado con el punto '.' de selección. Si se selecciona otro registro o si cambia el contenido del registro seleccionado, cambia también el puntero de memoria. La otra forma de escoger el valor del puntero de memoria es hacerlo directamente mediante el comando M o las flechas del cursor (estos comandos se explican más adelante).

Esto termina la descripción de ese tablero de control en que MON convierte la pantalla. Las dos secciones siguientes se dedican a explicar los comandos de MON. La sección 2 se dedica a los comandos habituales y la sección 3 a otros más avanzados. Procure aprender a manejar los primeros; cuando los conozca, verá seguramente que puede utilizar MON usando sólamente esos comandos.

SECCIÓN 2. COMANDOS HABITUALES

MON posee una gran cantidad de comandos posibles. Puede introducirse un comando en cuanto aparece la señal '⟩' de petición de comando en la parte superior izquierda de la pantalla. Realizaremos enseguida la descripción de cada comando, pero antes es necesario saber cómo introducir datos numéricos.

2.1 Introducción de números

En muchas ocasiones es necesario proporcionarle a MON números; por ejemplo, cuando se le deba dar una dirección.

Se pueden introducir números en forma decimal y hexadecimal. Un número en forma decimal debe ir precedido del símbolo '$'.

Los números decimales que se introduzcan deben constar de a lo sumo 5 dígitos, y los hexadecimales, de 4. Si se introducen más cifras, entonces se pierden las más significativas hasta quedarse con la cantidad permitida. Por ejemplo, si se pretende introducir $123456, lo que quedará en pantalla es $23456 y éste es el número que será realmente introducido.

Se pueden introducir números negativos, anteponiéndoles un signo menos. Recuérdese que cualquier número que se introduzca se toma módulo 65536 o 256, dependiendo de si el contexto permite palabras (16 bits) o bytes, respectivamente.

Para completar la introducción de un número basta con pulsar cualquier carácter que carezca de validez en el contexto del número. Para los números decimales, esto quiere decir cualquier carácter salvo Ø . . . 9; para los hexadecimales, cualquiera salvo Ø . . . 9 y A . . . F. Si el carácter que se pulsa es el de uno de los comandos válidos, el comando se ejecutará.

Es posible anular la entrada de un número pulsando ⟨CTRL/STOP⟩.

2.2 Descripción de los comandos

comando M acción: selecciona el puntero de memoria

Al pulsarlo, aparecerá una coma en la pantalla; eso significa que se debe introducir la dirección de memoria deseada. El puntero de memoria cambia a esa dirección, y alrededor de ella se centra el panel que muestra el contenido de la memoria. En cualquier momento se puede anular el comando pulsando ⟨CTRL/STOP⟩.

comando → acción: incrementa en 1 el puntero de memoria

Avanza en 1 el puntero y el panel de memoria.

comando ← acción: decrementa en 1 el puntero de memoria

Retrocede en 1 el puntero y el panel de memoria.

Durante la búsqueda aparece en pantalla el mensaje 'Wait...' (espere).

En cualquier momento puede anular el comando pulsando dos veces seguidas ⟨CTRL/STOP⟩.

comando N acción: busca de nuevo

Busca una nueva aparición de la sucesión de bytes que se han introducido con el comando 'G', lo que posibilita búsquedas repetidas. Cada vez que se produce un hallazgo, el panel de contenido de la memoria pasa a centrarse alrededor de esos bytes.

comando S acción: cambia el tipo de panel de contenido de memoria

Se emplea para alternar entre dos modelos diferentes de panel de contenido de la memoria. El primer modelo es el que aparece de entrada y que se ha explicado ya. El segundo es de la forma

```
 SP    IY    IX    HL    DE    BC    AF
DB85 ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ

3297 F3 s F3 s F3 s F3 s F3 s F3 s
F427 C3 C C3 C C3 C C3 C C3 C C3 C
4C73 D7 W D7 W D7 W D7 W D7 W D7 W
ØØØ6 B7 ? B7 ? B7 ? B7 ? B7 ? B7 ?
4891 Ø2 . Ø2 . Ø2 . Ø2 . Ø2 . Ø2 .
Ø8AA 1B . 1B . 1B . 1B . 1B . 1B .
8ØØ6 98 . 98 . 98 . 98 . 98 . 98 .
46Ø1 98 . 98 . 98 . 98 . 98 . 98 .
```

Está dividido en siete columnas, cada una correspondiente a un registro o par de registros. En la parte superior se muestra el contenido del registro. Más abajo aparece el contenido de las 8 primeras palabras (en el caso de SP), o de los primeros 8 bytes (en los demás casos) que se encuentran a partir de la dirección que contiene el registro. En el segundo caso se muestra a la derecha el equivalente ASCII de cada byte.

Este tipo de panel resulta útil para inspeccionar el contenido de la pila o de una tabla de valores cuya dirección está en algún registro.

comando O acción: salto relativo del puntero de memoria

Mueve el centro del panel de memoria como si éste efectuase un salto relativo cuya cantidad se encontrase en el byte al que señala el puntero de memoria. Se emplea para reproducir en el panel de contenido de memoria el salto que realizan las instrucciones 'JR' y 'DJNZ' del Z8Ø.

Supongamos que el puntero de memoria señala la dirección # 88ØØ y que las posiciones # 87FF y # 88ØØ contienen los bytes # 2Ø y # 16. Estos dos bytes pueden ser interpretados como una instrucción 'JR NZ,24' (o 'JR NZ, # 18'). Puede entonces ser interesante saber a dónde llevaría dicho salto. Esto es lo que realiza el comando 'O' cuando desplaza el centro del panel a la posición # 8817, que sería el destino del salto.

El comando 'O' trata los desplazamientos relativos superiores a # 7F (127) como desplazamientos negativos, en la misma forma que lo hace el Z8Ø.

comando U acción: inverso del comando 'O'

Devuelve el panel de contenido de memoria a la posición que tenía antes de efectuarse el comando 'O'.

comando X acción: salto absoluto del puntero de memoria

Hace que el puntero de memoria se traslade a la dirección contenida en la posición actual de dicho pun-

tero (la dirección contenida en el byte que señala el puntero y el siguiente), con lo que se modifica la zona que abarca el panel de memoria. Se emplea para reproducir en este panel el salto que realizan instrucciones como 'JP' o 'CALL'.

Supongamos que el puntero de memoria apunta al byte #Ø5 de la sentencia dada por los bytes #CD #Ø5 #83, que es 'CALL #83Ø5'. Entonces, mediante el comando 'X', el panel quedará centrado en la posición #83Ø5.

comando V acción: inverso del comando 'X'

Devuelve el panel de contenido de memoria a la posición que tenía antes de efectuarse el comando 'X'.

comando ⟨CTRL/X⟩ acción: devuelve el control a BASIC

Vuelve al intérprete BASIC. Si desde BASIC se desea volver a MON, se debe escribir

```
DEFUSR=&HB8ØØ ⟨RETURN⟩
A=USR(Ø) ⟨RETURN⟩
```

2.3 Cuadro resumen

Acabamos de ver los principales comandos utilizables para la corrección y depuración de programas. Existen otros, que veremos en la sección siguiente, pero normalmente bastará con utilizar los que ya hemos explicado. Como son bastante numerosos, vamos a dar un resumen de estos comandos, clasificados por el tipo de función que realizan.

Modificación del puntero de memoria

comando M	acción:	selecciona el puntero de memoria
comando →	acción:	incrementa en 1 el puntero de memoria
comando ←	acción:	decrementa en 1 el puntero de memoria
comando ↓	acción:	incrementa en 8 el puntero de memoria
comando ↑	acción:	decrementa en 8 el puntero de memoria
comando O	acción:	salto relativo del puntero de memoria
comando U	acción:	inverso del comando 'O'
comando X	acción:	salto absoluto del puntero de memoria
comando V	acción:	inverso del comando 'X'

Modificación del contenido de la memoria

comando I	acción:	copia un bloque de memoria
comando P	acción:	rellena un bloque de memoria

Además, si se teclea un número decimal (precedido de '$') o hexadecimal y a continuación se pulsa una tecla que no corresponda a una cifra válida ni a '.' (normalmente ⟨RETURN⟩ o las flechas del cursor), la posición de la RAM correspondiente al puntero de memoria se carga con dicho número.

Búsqueda

comando G	acción:	busca una sucesión de bytes
comando N	acción:	busca de nuevo

Modificaciones en los registros

comando . acción: avanza el punto '.' de selección de registro
comando R acción: carga un registro con el puntero de memoria

Además, si se teclea un número decimal (precedido de '$') o hexadecimal y a continuación el punto '.', el registro seleccionado se carga con dicho número.

Ejecución paso a paso de programas

comando Z acción: ejecuta programas paso a paso
comando J acción: salto a la dirección del PC
comando L acción: desensambla a partir del PC

comando O acción: salto relativo del puntero de memoria
comando U acción: inverso del comando 'O'
comando X acción: salto absoluto del puntero de memoria
comando V acción: inverso del comando 'X'

Varios

comando S acción: cambia el tipo de panel de contenido de memoria
comando ⟨CTRL/X⟩ acción: devuelve el control a BASIC

SECCIÓN 3. COMANDOS AVANZADOS

Los 9 comandos que vamos a ver en esta sección mejoran las prestaciones que proporciona MON. Es posible que usted no los considere necesarios. Le recomendamos, sin embargo, que lea al menos una vez esta sección cuando ya domine los comandos básicos.

3.1 Más sobre cómo cargar el programa

En la sección 1 nos limitamos a enseñarle a cargar MON en una dirección de memoria concreta, que era la #B800.

De hecho, MON se puede cargar en cualquier parte de los 32K altos del mapa de memoria del MSX, siempre que no entre en conflicto con el área de trabajo del MSX o la pila. En la práctica, esto significa que se lo puede cargar desde la posición #8100 hasta la #D400, para sistemas con magnetófono.

Para cargar MON debe teclear

BLOAD "CAS:MONMSX",&HXXXX

donde XXXX es la diferencia entre la dirección en que desea que se cargue y se ejecute MON y el número #8800. Por ejemplo, si se desea cargarlo en la posición #D000, se debe escribir

BLOAD "CAS:MONMSX",&H4800 ⟨RETURN⟩

A continuación se puede lanzar el programa mediante

DEFUSR=&HD000 ⟨RETURN⟩
A=USR (&HD000) ⟨RETURN⟩

Para cargarlo en #8100, que es inferior a #8800, habría que hacer

BLOAD "CAS:MONMSX",&HF900 ⟨RETURN⟩
DEFUSR=&H8100 ⟨RETURN⟩
A=USR(&H8100) ⟨RETURN⟩

Para volver a MON desde BASIC, lo que se debe teclear es

DEFUSR=&HXXXX ⟨RETURN⟩
A=USR(0) ⟨RETURN⟩

siendo XXXX la dirección de comienzo de MON.

3.2 Comandos para desensamblar

comando ⟨CTRL/L⟩ acción: desensambla a partir del puntero de memoria

Inicializa el panel de desensamblado en la dirección del puntero de memoria. Sirve para hurgar en la memoria y leer al mismo tiempo el código nemotécnico correspondiente.

Para volver a desensamblar a partir de la dirección del PC, utilice el comando 'L'.

comando ⟨CTRL/N⟩ acción: desensambla la siguiente página

Desensambla y muestra en el panel las siguientes 11 instrucciones.

Para volver a la 'página' inicial, pulse L o ⟨CTRL/L⟩, según convenga.

comando ⟨CTRL/D⟩ acción: desensambla un bloque de memoria

Desensambla un bloque de memoria; envía el correspondiente fichero de texto a la pantalla, a la impresora o a la cinta.

El comando empieza por pedir, mediante los mensajes 'First:' y 'Last:', las direcciones primera y última del bloque; éstas pueden ser introducidas en forma hexadecimal o decimal. Si la dirección final es menor que la inicial, el comando queda sin efecto.

A continuación aparece el mensaje 'Tape?', que le pregunta si desea grabar en cinta el fichero de salida. El fichero que se crea es un fichero de texto de los que puede cargar su ensamblador GEN. Para dar una respuesta afirmativa, pulse 'y ⟨RETURN⟩'; se le pedirá entonces el nombre del fichero; éste no debe tener más de 6 caracteres y debe ir seguido de ⟨RETURN⟩. Para dar una respuesta negativa, pulse simplemente ⟨RETURN⟩.

Luego le preguntará mediante el mensaje 'Printer?' si desea sacar el listado por la impresora. Si contesta 'y ⟨RETURN⟩', el listado irá a la impresora; si pulsa cualquier otra tecla, el listado irá a la pantalla.

Después aparece 'Workspace:'; el desensamblador le está pidiendo espacio para crearse una tabla de símbolos y un tampón para realizar la grabación en cinta (si debe hacerla). Para que utilice el que tiene previsto por defecto, pulse simplemente ⟨RETURN⟩; tomará entonces como área de trabajo la que comienza en uno de los tampones del generador de sonido (VOICAQ) y normalmente tendrá suficiente para la tabla de símbolos. Pero, si el bloque a desensamblar es mayor de 2K o si desea grabación en cinta (necesitará un tampón de 1K), tendrá que darle la dirección de un área de almacenamiento más grande. Como indicación, calcule 200 bytes por cada 1K que se desensamble y 1K para el tampón de cinta.

Finalmente, aparecerán repetidamente los mensajes 'First:' y 'Last:' para que delimite usted las secciones del bloque a desensamblar que deben ser interpretadas como áreas de datos, y no como instrucciones. El área que usted haya delimitado de esta manera será desensamblada en la forma 'DEFB XX', donde el contenido 'XX' de cada byte aparecerá como un carácter si su valor está entre 32 y 127, como un número hexadecimal en los demás casos. Cuando no desee delimitar más secciones de este tipo, responda con ⟨RETURN⟩ a los dos mensajes.

Terminada esta fase, la pantalla quedará en blanco durante el tiempo que tarde el desensamblador en dar una primera pasada para crear la tabla de símbolos. A continuación, si usted deseaba grabar el fichero en cinta, el mensaje 'Start tape, press any key' le pedirá que pulse una tecla cuando tenga lista la cinta en posición de grabación. Por fin, una segunda pasada del desensamblador producirá la aparición en la pantalla o la impresora del listado. Puede detener el listado con ⟨CTRL/STOP⟩. Entonces, pulsando de nuevo ⟨CTRL/STOP⟩, terminará el comando y la pantalla volverá a su contenido habitual. Pulsando cualquier otra tecla, el listado se reanudará.

Siempre que venga al caso (en las instrucciones de salto, por ejemplo), se crean etiquetas; son de la forma 'LXXXX', donde 'XXXX' es la dirección absoluta de la etiqueta en hexadecimal. Cuando esta dirección cae fuera de los límites del bloque que se trata, el desensamblador crea una línea con la seudo-operación 'EQU' para definir la etiqueta. De esta manera, el fichero de texto que se obtiene es conforme a las exigencias del ensamblador GEN.

Por ejemplo, si el comando se ha introducido en la forma:

⟩ ⟨CTRL/D⟩
First:Ø ⟨RETURN⟩
Last:1Ø ⟨RETURN⟩
Tape?: ⟨RETURN⟩
Printer?:y ⟨RETURN⟩
Workspace: ⟨RETURN⟩
First:4 ⟨RETURN⟩
Last:7 ⟨RETURN⟩
First: ⟨RETURN⟩
Last: ⟨RETURN⟩

aparecerá en la impresora el siguiente listado

```
ØØØØ F3             DI
ØØØ1 C3D7Ø2         JP    LØ2D7
ØØØ4 BF1B9898       DEFB #BF,#1B,#98,#98
ØØØ8 C38326         JP    L2683
ØØØB ØØ             MOP
ØØØC C3B6Ø1         JP    LØ1B6
ØØØG ØØ             NOP
ØØ1Ø C38626         JP    L2686
               LØ1B6
               LØ2D7
               L2683
               L2686
```

3.3 Comandos de interrupción

comando ⟨CTRL/B⟩ acción: dispone una interrupción

Dispone una interrupción en la dirección del puntero de memoria. El comando sustituye (y guarda) los 3 bytes que comienzan en esa dirección por una instrucción CALL. Cuando se ejecuta esta llamada, lo que hace es restituir los 3 bytes originales, lanzar un aviso sonoro para que usted pulse una tecla y, entonces, actualiza la pantalla con la situación del ordenador.

Es necesario ser cuidadoso con el lugar en que se coloca la interrupción ya que a veces pueden producirse efectos inesperados. Vamos a poner un ejemplo. Suponga que tiene en la memoria

... 2Ø Ø1 3C 21 44 98 ...

y que el puntero de memoria señala al byte 3C. Los bytes 3C 21 44 serán sustituidos por una sentencia CALL. Cuando se ejecute 2Ø Ø1, que es un salto relativo condicional JR NZ de 3 unidades, es posible que la condición sea cierta y que se efectúe el salto justamente a la mitad de la instrucción CALL, lo que producirá un resultado absolutamente imprevisible.

Utilice únicamente este comando cuando desee disponer varias interrupciones a lo largo de un programa. Lo normal es que le baste con el comando 'J' en los casos habituales.

comando ⟨CTRL/R⟩ acción: suprime las interrupciones

Suprime las interrupciones que se han introducido y devuelve al código su contenido original.

3.4 Comandos de ejecución

Si desea ejecutar paso a paso su programa, puede utilizar el comando 'J' que ya explicamos. Tiene también a su disposición los tres comandos que vamos a describirle. Dos de ellos (el ⟨CTRL/J⟩ y el ⟨CTRL/Z⟩) hacen posible la ejecución simulada del código, que es lenta pero tiene la ventaja de permitirle desmenuzar su programa para que pueda conocer el detalle de la ejecución. Durante esta ejecución puede usted detener el proceso con ⟨CTRL/STOP⟩ y ver entonces en la pantalla el estado del ordenador en el momento de la detención.

Para una ejecución en modo rápido, utilice el comando 'J' o el comando ⟨CTRL/S⟩.

comando ⟨CTRL/J⟩ acción: ejecución simulada hasta una interrupción

Produce la ejecución simulada de su programa desde la dirección del PC hasta que encuentre una interrupción. Puede usted disponer una interrupción en el lugar que desee introduciendo una dirección en respuesta al mensaje ':' que aparecerá en pantalla. Si no desea introducirla, pulse simplemente ⟨RETURN⟩.

La ejecución es en modo lento. Durante esta ejecución puede usted detener el proceso con ⟨CTRL/STOP⟩ y ver entonces en la pantalla el estado del ordenador en el momento de la detención.

comando ⟨CTRL/Z⟩ acción: ejecución simulada de un bucle

Produce la ejecución simulada (en modo lento) de un bucle. El número de veces que se desea repetirlo debe ser introducido cuando el comando lo solicite mostrando en pantalla el símbolo '#'; introduzca el número (en decimal o hexadecimal) y terminelo con ⟨RETURN⟩. La ejecución comenzará en la dirección del PC y continuará hasta que se haya pasado por esta misma dirección el indicado número de veces. Por ejemplo, supongamos que se tiene

```
8452 2B             DEC HL
8453 1Ø BB          DJNZ L84ØØ
8455 21 FF FF       LD HL,#FFFF
```

que el PC contiene #8452 y que el registro B contiene #2Ø. Se puede ejecutar 16 veces el bucle que termina con la sentencia 'DJNZ' (es decir, hasta que en B haya #1ØØ) introduciendo '1Ø ⟨RETURN⟩' o '$16 ⟨RETURN⟩' en respuesta al símbolo '#'.

La ejecución es en modo lento. Durante esta ejecución puede usted detener el proceso con ⟨CTRL/STOP⟩ y ver entonces en la pantalla el estado del ordenador en el momento de la detención.

comando ⟨CTRL/S⟩ acción: pasa rápidamente por una instrucción

Dispone una interrupción (en la forma habitual que hemos explicado en 'J') al final de la instrucción a la que apunta el PC y ejecuta en modo rápido dicha instrucción. Suele ser útil cuando se está siguiendo un programa en modo lento y se desea pasar rápidamente una instrucción cuya ejecución duraría demasiado (una sentencia 'CALL', por ejemplo).

De hecho, este comando equivale al 'J' con 'J:X + N' siendo X la dirección de la instrucción que señala el PC y N su longitud.

De nuevo le advertimos sobre los resultados imprevisibles que puede originar la alteración de 3 bytes que produce este comando.

3.5 Otro comando

comando ⟨CTRL/K⟩ acción: modifica la configuración de los canales (“slots”)

Le piede que introduzca un número de 8 bits y envía este número a la puerta #A8 para modificar la configuración de los canales de la máquina. Utilice este comando con cuidado, ya que puede desconectar la RAM en la que se encuentra cargado MON.

La configuración de canales en cada momento aparece en el panel de estado de los registros, a la izquierda de la posición que ocupa el contador del programa (PC).

Producido por:
indescomp
Av. del Mediterráneo, 9. 28007 MADRID.